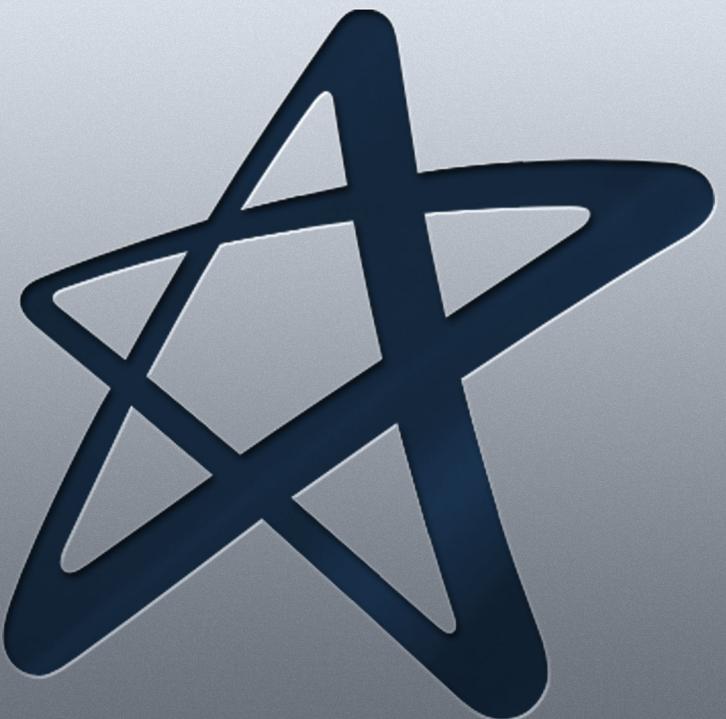


Banco de Dados



Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

Material Teórico



Procedures, Functions, Exceptions e Triggers

Responsável pelo Conteúdo:

Prof. Ms. Alexander Gobbato Albuquerque

Revisão Textual:

Prof. Ms. Claudio Brites

UNIDADE

Procedures, Functions, Exceptions e Triggers



- Introdução
- EXCEPTION
- Vantagens de Exceções PL / SQL
- Triggers



Objetivo de APRENDIZADO

O conceito de Stored Procedure é muito importante e grande parte do mundo corporativo utiliza esse recurso. Para que nossa abordagem seja ainda mais profissional, iremos estudar também as exceções que podemos criar em PL/SQL para tratamento de erros. Dedicaremos um tópico, ainda, aos Triggers, com os quais podemos invocar várias chamadas para executar uma determinada tarefa após o acontecimento de um evento.

Resumindo, nesta unidade trabalharemos os seguintes tópicos de conteúdo: PL/SQL; Procedures; Exceptions; Function; Triggers.

Para obter um bom aproveitamento da unidade, vamos conferir sua estrutura:

Atividade de Sistematização: os exercícios disponibilizados são de autocorreção e tem por objetivo te ajudar a avaliar o que aprendeu na disciplina, identificando se há pontos que precise rever, ou se há necessidade, ainda, de pedir auxílio ao tutor. Além disso, as notas atribuídas aos exercícios serão parte de sua média final da disciplina.

Atividade de Aprofundamento: Ver a atividade que será disponibilizada, ela também será avaliada e a nota irá compor sua média final.

Lembramos a você da importância de realizar todas as atividades propostas dentro do prazo estabelecido para cada unidade, dessa forma, você evitará que o conteúdo se acumule e que você tenha problemas ao final do semestre.

Uma última recomendação, caso tenha problemas para acessar algum item da disciplina, ou dúvidas em relação ao conteúdo, não deixe de entrar em contato com seu professor tutor por meio do link Mensagens.

Contextualização

Program Language SQL é a linguagem de programação da Oracle que objetiva processar informações do banco de dados.



O Stored Procedure é um subprograma que é compilado e armazenado no servidor. Ele pode ser chamado a partir de um comando SQL qualquer.

A vantagem de usar procedimentos armazenados é que eles podem encapsular rotinas de uso frequente no próprio servidor, e estarão disponíveis para todas as aplicações.

Vamos aprender?

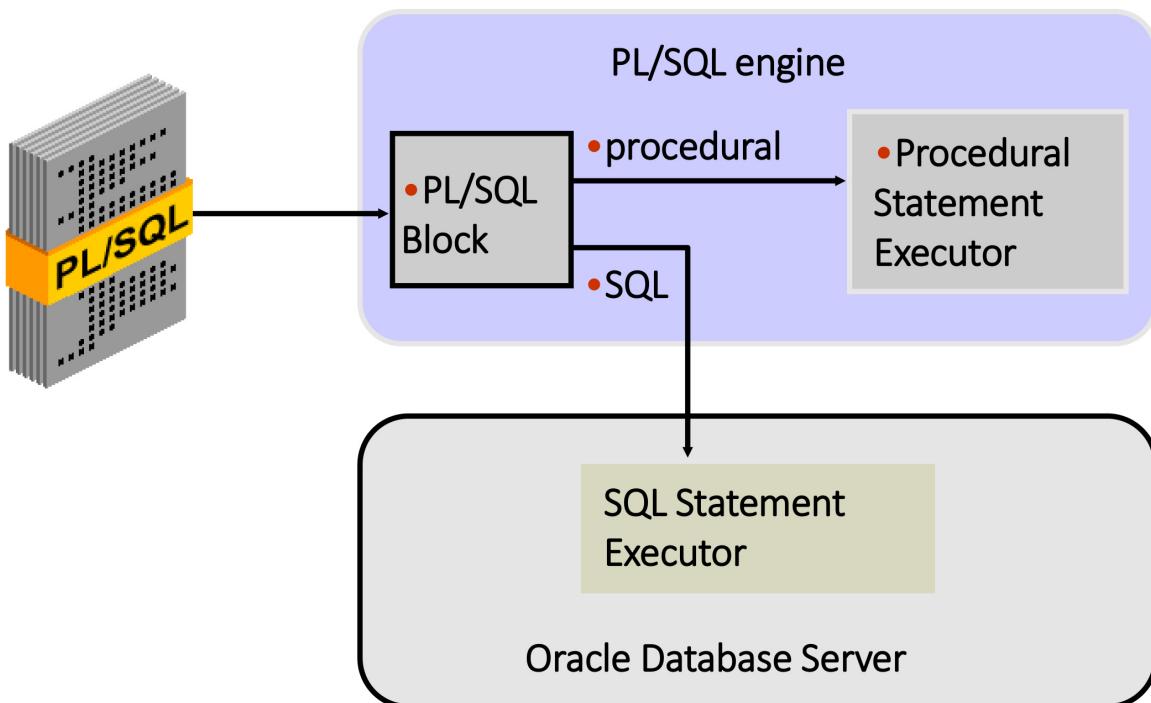
Introdução



Vamos relembrar que com PL/SQL podemos criar variáveis herdando o tipo e tamanho de outras variáveis ou objetos do banco de dados, tais como: tabelas; criar cursores para tratamento de resultados de queries; criar registros para armazenar resultado de cursores; tratamento de erros; e utilizar comandos de repetição e comparação.

Estudamos também anteriormente que Stored Procedure são subprogramas compilados e armazenados no servidor. Eles podem ser chamados a partir de um comando SQL qualquer.

Program Language SQL: é a linguagem de programação da Oracle que objetiva processar informações do banco de dados.



Stored Procedure

Relembrando que Stored Procedure são Blocos PL/SQL armazenados no banco de dados de forma compilada, podem ou não receber parâmetros. Os parâmetros passados para os subprogramas podem ser de 3 tipos:

- IN (padrão) – passa um valor do ambiente chamador para o subprograma. Esse valor não pode ser alterado dentro do subprograma;
- OUT – passa um valor do subprograma para o ambiente chamador;
- IN OUT – passa um valor do ambiente chamador para o subprograma. Esse valor pode ser alterado dentro do subprograma e retornado com o valor atualizado para o ambiente chamador.

Functions

São subprogramas que têm por objetivo retornar algum resultado ou retornar algum valor.
Estrutura para criação de function:

CREATE OR REPLACE FUNCTION Nome_Funçao

(Argumento1 modo Tipo_de_Dados,
Argumento2 modo Tipo_de_Dados,
Argumento3 modo Tipo_de_Dados)

 RETURN Tipo_de_Dado

IS ou AS

declarações

Begin

 Bloco PL/SQL

Exception

End nome_da_função;

A instrução FUNCTION é usada para criar uma função autônoma ou uma função específica. A função armazenada – também chamada de função do usuário ou função definida pelo usuário. A função é um conjunto de instruções PL/SQL chamada pelo nome. Funções armazenadas são muito parecidas com as PROCEDURES, com exceção de que uma função retorna um valor. Funções do usuário podem ser usadas como parte de uma expressão SQL.

As funções não podem ser usadas nas seguintes opções:

Em uma cláusula de restrição CHECK de um CREATE TABLE ou ALTER TABLE; em uma cláusula DEFAULT de uma instrução CREATE TABLE ou ALTER TABLE.

Além disso, quando uma função é chamada a partir de uma consulta ou instrução DML, a função não pode ter OUT ou IN OUT como parâmetros de informações.

Exemplo:

CREATE or REPLACE FUNCTION balanco (numcont IN NUMBER)

 RETURN NUMBER

 IS contabalanco NUMBER(11,2);

 BEGIN

 SELECT saldo_conta

 INTO contabalanco

```
FROM pedidos  
WHERE pedido_conta = numcont;  
RETURN(contabanco);  
END;
```

A função balanco retorna o saldo de uma conta especificada. Quando a função é executada, o argumento numcont recebe o número da conta e busca seu saldo. A função usa uma instrução SELECT para selecionar a coluna identificada pelo argumento numcont da tabela de pedidos. A função usa uma instrução RETURN para retornar esse valor para o ambiente em que ela foi chamada.

Ao especificar a sintaxe REPLACE, a função será recriada se ela já existir. Utilize essa cláusula para mudar a definição de uma função existente, sem alterar os privilégios. Se você redefinir uma função, então o Oracle Database irá recompilá-lo.

Para testarmos uma função, podemos executar a seguinte instrução:

```
SELECT <nome_função> FROM dual.
```

Utilizando como exemplo a função criada acima a sintaxe seria:

```
SELECT balanco(12345) from dual
```

EXCEPTION



São todos os erros e imprevistos que podem ocorrer durante a execução de um bloco PL/SQL.

EXCEPTION

```
WHEN nome_da_exceção THEN
```

Relação_de_comandos

```
WHEN nome_da_exceção THEN
```

Relação_de_comandos

Vantagens de Exceções PL / SQL



Ao usarmos exceções para manipulação de erros, temos várias vantagens. Com as exceções é possível tratar os erros de forma confiável com um manipulador de exceção única:

```

BEGIN
    SELECT ...
    SELECT ...
    procedure_that_performs_select();
    ...
EXCEPTION
    WHEN NO_DATA_FOUND THEN -- catches all 'no data found' errors

```

Ao invés de verificar se existe um erro em um determinado ponto, adicionaremos um manipulador de exceção para o bloco PL/SQL. Se a exceção é sempre executada nesse bloco (ou qualquer sub-bloco), teremos certeza que será tratada.

Às vezes, o erro não é muito óbvio e não pode ser detectado tão facilmente. Com um único tratamento de exceção podemos capturar cálculos com dados incorretos, erros de divisão por zero e assim por diante. Se houver a necessidade de realizar um tratamento em um ponto específico, podemos colocar uma única instrução ou um grupo de instruções dentro de seu próprio bloco BEGIN-END.

Existem exceções pré-definidas no Oracle como, por exemplo:

Oracle Exception Name	Oracle Error	Explanation
DUP_VAL_ON_INDEX	ORA-00001	You tried to execute an INSERT or UPDATE statement that has created a duplicate value in a field restricted by a unique index.
TIMEOUT_ON_RESOURCE	ORA-00051	You were waiting for a resource and you timed out.
TRANSACTION_BACKED_OUT	ORA-00061	The remote portion of a transaction has rolled back.
INVALID_CURSOR	ORA-01001	You tried to reference a cursor that does not yet exist. This may have happened because you've executed a FETCH cursor or CLOSE cursor before OPENing the cursor.
NOT_LOGGED_ON	ORA-01012	You tried to execute a call to Oracle before logging in.
LOGIN_DENIED	ORA-01017	You tried to log into Oracle with an invalid username/password combination.

NO_DATA_FOUND	ORA-01403	You tried one of the following: 1. You executed a SELECT INTO statement and no rows were returned. 2. You referenced an uninitialized row in a table. 3. You read past the end of file with the UTL_FILE package.
TOO_MANY_ROWS	ORA-01422	You tried to execute a SELECT INTO statement and more than one row was returned.
ZERO_DIVIDE	ORA-01476	You tried to divide a number by zero.
INVALID_NUMBER	ORA-01722	You tried to execute an SQL statement that tried to convert a string to a number, but it was unsuccessful.
STORAGE_ERROR	ORA-06500	You ran out of memory or memory was corrupted.
PROGRAM_ERROR	ORA-06501	This is a generic "Contact Oracle support" message because an internal problem was encountered.
VALUE_ERROR	ORA-06502	You tried to perform an operation and there was an error on a conversion, truncation, or invalid constraining of numeric or character data.
CURSOR_ALREADY_OPEN	ORA-06511	You tried to open a cursor that is already open.

No exemplo a seguir, usaremos uma exceção já criada no Oracle:

```
set serveroutput on
Begin
Insert into Pais values(&codigo,'&nome');
Commit;
Dbms_output.put_line('Comando executado com sucesso');
```

Exception

When Dup_Val_On_Index Then

```
Dbms_output.put_line ('ERRO!!! Código de país já existente');
```

When Others then Dbms_output.put_line ('Erro ao cadastrar o país');

```
end;
```

Também é possível criar exceções diferentes das que já foram definidas e criadas pelo sistema de banco de dados. A exceção que foi criada pelo usuário deverá ser invocada pelo comando RAISE, exemplo:

Declare

Nome_da_exceção EXCEPTION;

Begin

Relação_de_comandos

IF ... Then

RAISE Nome_da_exceção;

End If;

Relação_de_comandos

EXCEPTION

WHEN nome_da_exceção THEN

Relação_de_comandos

End;

Vejamos um exemplo prático de um tratamento de exceção criado pelo usuário:

SET SERVEROUTPUT ON

SET VERIFY OFF

declare

```
codigo cliente.cd_cliente%type;
nome cliente.nm_cliente%type;
fisica cliente.ie_fisica_juridica%type;
sexo cliente.ie_sexo%type;
est_civ cliente.est_civ%type;
Valida_campos exception;
```

Begin

codigo:=&codigo;

nome:='&nome';

fisica:='&fisica';

sexo:='&sexo';

est_civ:='&est_civ';

If (fisica='F' and (sexo is null or est_civ is null)) then

```

Raise Valida_Campos;

Elseif (fisica= 'J' and (sexo is not null or est_civ is not null)) then

    Raise Valida_campos;

Else

    Insert into cliente (cd_cliente, nm_cliente, ie_fisica_juridica, ie_sexo, est_civ)
    values (codigo,nome, fisica, sexo, est_civ);
    dbms_output.put_line ('Cliente cadastrado com sucesso');

    COMMIT;

end if;

Exception

    When Valida_campos then
        dbms_output.put_line ('Verificar campos sexo ou estado civil preenchido incorretamente');

    when Dup_Val_On_Index then
        dbms_output.put_line (' ERRO!!! Código já cadastrado, favor cadastrar novo código
diferente de '|| codigo);

    end;

```

Triggers



São blocos PL/SQL disparados automaticamente e implicitamente sempre que ocorrer um evento associado a uma determinada tabela (INSERT, UPDATE ou DELETE).

Um Trigger pode ser utilizado para:

- Manutenção de tabelas de auditoria;
- Manutenção de tabelas duplicadas;
- Implementação de níveis de segurança mais complexos;
- Geração de valores de colunas – por exemplo, gerar o valor total da nota fiscal a cada inclusão, exclusão ou alteração de itens da nota.

Por exemplo, podemos criar um Trigger – ou gatilho, como também é conhecido – para executar a seguinte tarefa: ao se alterar um valor da tabela produto, automaticamente grava-se o valor antigo e o novo valor em uma tabela chamada de Tmp_Preco_Prod.

Tabela produto:

cd_produto	vl_custo_medio
1	1.5
2	2.3
3	.65
4	3.12
5	1.5

Atualizar o vl_custo_medio do produto de código 4 para 2.93

cd_produto	vl_anterior	vl_novo
4	3.12	2.93

Ao executar o comando *Update produto set vl_custo_medio =2.93 where cd_produto=4*, a tabela *tmp_preco_prod* é alimentada automaticamente.

Para se criar um trigger a sintaxe é a seguinte:

CREATE OR REPLACE TRIGGER BASICO (1)
BEFORE (2) UPDATE (3) ON AUDITORIA (4)
FOR EACH ROW
[BLOCO PL/SQL]

Onde (1) é o nome de seu *Trigger*, (2) é o tempo em que será disparado o evento – BEFORE OU AFTER – (3) será o comando DML que será executado no sistema – pode ser UPDATE, INSERT ou DELETE – e (4) é o nome da tabela que você quer que um trigger seja executado.

Create or Replace Trigger Verifica_Produto

Before Update of vl_custo_medio on produto

For each row

Begin

Insert into Tmp_Precio_Prod

Values(:old.cd_produto, :old.vl_custo_medio, :new.vl_custo_medio);

End;

Entendendo os pseudo registros :Old e :New

Ao trabalharmos com Triggers de nível de linha, utilizamos estes dois pseudo registros, eles servem para fazer as comparações das colunas velhas (:old) com as novas (:new), são muito utilizados para fazer Update nas colunas.



Os registros especiais :NEW e :OLD armazenam temporariamente os valores do último registro manipulado pelo SGBD.

Ao inserir, é criado o registro :NEW

Ao apagar, é criado o registro :OLD

Ao atualizar, são criados os 2 registros :NEW e :OLD

DML	:OLD	:NEW
INSERT	NULO	VALORES NOVOS
DELETE	VALORES ANTIGOS	NULO
UPDATE	VALORES ANTIGOS	VALORES NOVOS

Desabilitando e Habilitando um Trigger Específico:

```
ALTER TRIGGER nome_da_trigger DISABLE;
```

```
ALTER TRIGGER nome_da_trigger ENABLE;
```

Desabilitando e Habilitando todos os Triggers de uma tabela:

```
ALTER TABLE nome_tabela DISABLE ALL TRIGGERS;
```

```
ALTER TABLE nome_tabela ENABLE ALL TRIGGERS;
```

Eliminando Triggers e Procedimentos:

```
DROP TRIGGER nome_trigger;
```

```
DROP PROCEDURE nome_procedure;
```

Inspecionando o dicionário de dados:

```
SELECT object_name, object_type FROM user_objects WHERE object_type ='TRIGGER';
```

Este comando é executado para ver todas as triggers do Usuário.

Obtendo detalhes dos Triggers:

```
DESC USER_TRIGGERS
```

Este comando lhe dará todas as informações objetivas dos triggers, como: tabela associada, corpo, Variável utilizada, etc.

Material Complementar

O conteúdo sobre Function e Trigger pode ser muito mais aprofundado, vale a pena dar uma lida na documentação da Oracle sobre essas duas técnicas. Disponível em:

- http://docs.oracle.com/cd/B19306_01/appdev.102/b14251/adfns_triggers.htm

E também:

- <http://www.techonthenet.com/oracle/functions.php>

Referências

Fanderuff, Damaris. **Dominando o Oracle 9i: modelagem e desenvolvimento.** São Paulo: Pearson Education do Brasil, 2003.

Material oficial da Oracle no curso: Oracle Database 10 G – SQL Fundamentals I.

Sobre Function: http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5009.htm#i2092700

Sobre Triggers: http://docs.oracle.com/cd/B19306_01/appdev.102/b14251/adfns_triggers.

Anotações





Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

www.cruzeirodosulvirtual.com.br
Campus Liberdade
Rua Galvão Bueno, 868
CEP 01506-000
São Paulo SP Brasil
Tel: (55 11) 3385-3000



Universidade
Cruzeiro do Sul



UNICID
Universidade
Cidade de S. Paulo



UNIFRAN
Universidade
de Franca



UDF
Centro
Universitário



Módulo
Centro
Universitário